

## PATENT ABSTRACTS OF JAPAN

5

(11)Publication number : 05-066954

(43)Date of publication of application : 19.03.1993

(51)Int.Cl.

G06F 9/46

G06F 15/00

(21)Application number : 03-252909

(71)Applicant : KOBE NIPPON DENKI SOFTWARE  
KK

(22)Date of filing : 05.09.1991

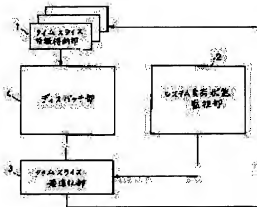
(72)Inventor : SHIBANO RYOICHI

## (54) TIME SLICE OPTIMIZATION SYSTEM

## (57)Abstract:

PURPOSE: To facilitate the effective operation of a time slice optimization system by adjusting automatically the time slice value of each program in accordance with the system state.

CONSTITUTION: A time slice information storing part 1 stores the time slice information on each program. A dispatching part 4 dispatches a CPU to each program based on the time slice value stored in the part 1. A system load state monitoring part 2 monitors the load state of the system. Then a time slice optimizing part 3 optimizes the time slice value of each program stored in the part 1 based on the information that is carried out through the part 4 at a fixed interval and acquired through the part 2.



## LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]

(19)日本国特許庁(J P)

(12) 公開特許公報(A)

(11)特許出願公開番号

特開平5-66954

(43)公開日 平成5年(1993)3月19日

(51)Int.Cl. <sup>3</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 9/46	3 4 0 E	8120-5B		
		D 8120-5B		
15/00	3 1 0 K	8219-5L		

審査請求 未請求 請求項の数1(全 9 頁)

(21)出願番号 特願平3-252909

(22)出願日 平成3年(1991)9月5日

(71)出願人 000192545

神戸日本電気ソフトウェア株式会社  
兵庫県神戸市西区高塚台5丁目3番1号

(72)発明者 芝野 良一

兵庫県神戸市西区高塚台5丁目3番1号  
神戸日本電気ソフトウェア株式会社内

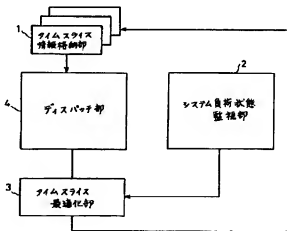
(74)代理人 弁理士 山川 政樹

(54)【発明の名称】 タイムスライス最適化方式

(57)【要約】

【目的】 各プログラムのタイムスライス値をシステムの状態に合わせて自動的に調整を行い、容易に効率的なシステム運用が行えるようにする。

【構成】 タイムスライス情報格納部1により各プログラムのタイムスライス情報を格納し、ディスパッチ部4でこのタイムスライス情報格納部1に格納されているタイムスライス値を基に各プログラムにCPUをディスパッチし、システム負荷状態監視部2でシステムの負荷状態を監視し、タイムスライス最適化部3によりディスパッチ部4から一定間隔で実行されシステム負荷状態監視部2から得た情報を基にタイムスライス情報格納部1に格納している各プログラムのタイムスライス値を最適化する構成にした。



## 【特許請求の範囲】

【請求項1】 各プログラム毎に異なるタイムスライズ値を用いてCPUディスパッチを行うマルチプログラミングシステムにおいて、現在実行している各プログラム毎にそのプログラムに関するタイムスライズ情報を格納するタイムスライズ情報格納部と、一定間隔でシステムの負荷状況を監視しているシステム負荷状態監視部と、一定期間毎に前記システム負荷状態監視部より取得したその期間のシステム負荷状態より前記タイムスライズ情報格納部に格納した各プログラムのタイムスライズ値を最適化するタイムスライズ最適化部と、このタイムスライズ最適化部で最適化して前記タイムスライズ情報格納部に格納しているタイムスライズ値で前記CPUをプログラムにディスパッチするディスパッチ部を設け、システムの状態に合わせて自動的にタイムスライズ値を調整するようにしたことを特徴とするタイムスライズ最適化方式。

## 【発明の詳細な説明】

## 【0001】

【産業上の利用分野】本発明は各プログラム毎に異なるタイムスライズを用いてCPUディスパッチを行うマルチプログラミングシステムにおいて、システムの負荷状態に合わせて自動的にタイムスライズ値の調整をすることができるタイムスライズ最適化方式に関するものである。

## 【0002】

【従来の技術】従来、マルチプログラミングシステムでは、特定のプログラムが長時間CPUを占有することを防ぐためにタイムスライズを用いたCPUディスパッチを行っていた。このタイムスライズはプログラム毎または複数のプログラムからなるプログラムグループ毎に設定される。そして、一般に計算処理を主体とするプログラムの場合はディスパッチされたCPUをタイムスライズを使いきって放棄することが多く、逆にインタラクティブ処理や入出力処理が主体のプログラムの場合はディスパッチされたCPUを入出力の完了待ちやその他のイベント待ち等の理由でタイムスライズを使いきらずに放棄することが多い。また、CPUの使用が放棄された時点でディスパッチング処理が実行されるシステムのオーバヘッドの一部となるため、システムの使用効率をより高めるためにはタイムスライズの値を大きくし単位時間当たりのディスパッチング処理の実行回数を減らすことが有用である。

【0003】しかし、各プログラムのタイムスライズ値を大きくすると、システムの負荷が軽い場合は特に問題にはならないが、システムの負荷が高くなるとインタラクティブ処理を主体とするプログラムの応答性が極端に遅くになってしまう。したがって、システムの負荷が軽い間は長めのタイムスライズ値が有効であり、負荷が高くなると短めのタイムスライズ値が有効となる。

## 【0004】

【発明が解決しようとする課題】上述した従来の方式では、各プログラムに設定するタイムスライズは固定であり、システムの状態に合わせて自動的に変更することはできないという課題があった。

## 【0005】

【課題を解決するための手段】本発明のタイムスライズ最適化方式は、各プログラム毎に異なるタイムスライズ値を用いてCPUディスパッチを行うマルチプログラミングシステムにおいて、現在実行している各プログラム毎にそのプログラムに関するタイムスライズ情報を格納するタイムスライズ情報格納部と、一定間隔でシステムの負荷状況を監視しているシステム負荷状態監視部と、一定期間毎に上記システム負荷状態監視部より取得したその期間のシステム負荷状態より上記タイムスライズ情報格納部に格納した各プログラムのタイムスライズ値を最適化するタイムスライズ最適化部と、このタイムスライズ最適化部で最適化して上記タイムスライズ情報格納部に格納しているタイムスライズ値で上記CPUをプログラムにディスパッチするディスパッチ部を設け、システムの状態に合わせて自動的にタイムスライズ値を調整するようにしたものである。

## 【0006】

【作用】本発明においては、一定期間毎にその間のシステム負荷状態により各プログラムのタイムスライズ値を見直し、最適なシステム運用が行えるように自動的にタイムスライズ値の調整を行うようにする。

## 【0007】

【実施例】図1は本発明の一実施例を示すブロック図である。この図1において、1は現在実行している各プログラム毎にそのプログラムに関するタイムスライズ情報を格納するタイムスライズ情報格納部、2は一定間隔でシステムの負荷状況を監視しているシステム負荷状態監視部、3は一定期間毎にシステム負荷状態監視部2より取得したその期間のシステム負荷状態よりタイムスライズ情報格納部1に格納した各プログラムのタイムスライズ値を最適化するタイムスライズ最適化部、4はこのタイムスライズ最適化部3で最適化してタイムスライズ情報格納部1に格納しているタイムスライズ値でCPUをプログラムにディスパッチするディスパッチ部である。

【0008】ここで、上記システム負荷状態監視部2は周期的にシステムの負荷状況を監視しているシステム負荷状態監視部であり、タイムスライズ最適化部3は一定期間毎にシステム負荷状態監視部2より得る現在のシステムの負荷状況によって各プログラムに設定したタイムスライズ値に対する調整値を計算しタイムスライズ情報格納部1に格納された各プログラムのタイムスライズ値を調整するタイムスライズ最適化部である。また、ディスパッチ部4は次に実行すべきプログラムを決定しそのプログラムに対しタイムスライズ情報格納部1に格納さ

3

れたタイムスライズ値でCPUをディスパッチするとともに一定期間毎にタイムスライズ最適化部3を実行するディスパッチ部である。そして、システムの状態に合わせて自動的にタイムスライズ値を調整するように構成されている。

【0009】図2は図1におけるタイムスライズ情報格納部1の構成例を示す説明図である。この図2において、11はこのプログラムに最初に設定されたタイムスライズ初期値を示し、12は現在のタイムスライズ値、13は取り得ることのできるタイムスライズ値の最大値、14はタイムスライズ値の最小値を示す。そして、これらの情報は対応するプログラムが実行可能になった時点で新しく用意され値が格納される。その場合、現在のタイムスライズ値12の値はタイムスライズ初期値11と同じ値が格納される。

【0010】図3は図1におけるディスパッチ部4のアルゴリズムの例を示すフローチャートである。まずシステム起動時にディスパッチ部4がタイムスライズ最適化部3を実行する間隔を設定する(ステップ101)。つぎに、実行すべきプログラムを決定し、そのプログラムのタイムスライズ情報を格納しているタイムスライズ情報格納部1を参照し、図2に示す現在のタイムスライズ値12だけそのプログラムにCPUをディスパッチする(ステップ102〜104)。そして、そのプログラムがCPUの使用を放棄すると、ディスパッチ部4は以前タイムスライズ最適化部3を実行してから設定した期間が過ぎたかどうかを調べる(ステップ105、106)。過ぎていた場合にはタイムスライズ最適化部3を実行し、次にタイムスライズ最適化部3を実行するまでの期間を設定する(ステップ107、108)。その後、ディスパッチ部4は次に実行するプログラムの選定に戻る。

【0011】図4は図1におけるタイムスライズ最適化部3のアルゴリズムの例を示すフローチャートである。まず、タイムスライズ最適化部3はディスパッチ部4により一定期間毎に実行される。そして、タイムスライズ最適化部3はシステム負荷状態監視部2より前回調べたから今までのシステムの負荷状況を取得する。この例ではシステムの負荷状況としてその期間中のアイドルCPU比率を取得している(ステップ201)。つぎに、このシステム負荷状況よりタイムスライズ値の調整を行うべきか否か、または調整する場合はその調整値をどのくらいにするかを決定する。この例の場合はアイドルCPU比率が25%以下の場合にはシステムの負荷が高いとみなし(ステップ202)、次回からのタイムスライズ値を減らすために調整値としてマイナスの値(−△)を設定する(ステップ203)。逆にアイドルCPU比率が75%以上の場合(ステップ204)はシステムの負荷が軽いとみなし、次回からのタイムスライズ値を増やすために調整値としてプラスの値(+△)を設定する(ス

4

テップ205)。ここで、アイドルCPU比率が25%以下でも85%以上でもない場合には、現在のタイムスライズ値が妥当であるとみなし、タイムスライズ値の調整は行わない。

【0012】つぎに、各実行中のプログラムに対応するタイムスライズ情報格納部1を順番に調べ(ステップ206)、タイムスライズ情報格納部1内の現在のタイムスライズ値12の値に上記調整値を加える(ステップ207)。ただし、加算した後の値が上記タイムスライズ情報格納部1内のタイムスライズ値の最大値13より大きくなった場合は(ステップ208)、現在のタイムスライズ値12の値をタイムスライズ値の最大値13にする(ステップ209)。これはタイムスライズ値が無限に大きくなるのを防ぐためである。また、加算した後の値が上記タイムスライズ情報格納部1内のタイムスライズ値の最小値14より小さくなった場合には(ステップ210)、現在のタイムスライズ値12の値をタイムスライズ値の最小値14にする(ステップ211)。これはタイムスライズ値が「0」以下になることを防ぐためである。

【0013】そして、上記調整値の決め方は図4の例の方法の他に図5に示すような表を用いて行うこともできる。図5は図1におけるタイムスライズ最適化部3で使用する調整値決定用の表の一例を示す説明図である。この場合はシステム負荷状態監視部2より得たアイドルCPU比率で表の条件部を検索し、合致した条件部の横の調整値をタイムスライズの調整値とするともに、その値が「0」の場合は調整の必要がないものとする。

【0014】図6は図1におけるシステム負荷状態監視部2の構成例を示すブロック図である。システム負荷状態監視部2は、一定時間間隔で割り込みを発生させるタイマ部21と、このタイマ部21の発生する割り込み毎にシステムの負荷状態を調べるシステム状態調査部22と、このシステム状態調査部22の結果を記録保存するシステム状態記録部24と、タイムスライズ最適化部3からの状態問い合わせに対してシステム状態記録部24の内容より現在のシステムの負荷状態を答えるシステム状態回答部23から構成されている。

【0015】図7はこの図6におけるシステム状態調査部22のアルゴリズムの例を示すフローチャートである。システム状態調査部22は、起動時にまず、システム状態記録部24内の各値を「0」にクリアし(ステップ301)、つぎに、システム状態を監視する間隔を示す割り込み間隔をタイマ部21に設定する(ステップ302)。タイマ部21からの割り込みが発生する毎にCPUの状態を調べ(ステップ303、304)、CPUが使用中であれば(ステップ305)、システム状態記録部24内のCPU使用回数に「1」足す(システム306)。使用中で無い場合には同じくシステム状態記録部24内のCPUアイドル回数に「1」足す(ステッ

ブ307)。その後再度割り込み待ちに戻る。

【0016】図8は図6におけるシステム状態回答部23のアルゴリズムの例を示すフローチャートである。まず、システム状態回答部23は、タイムスライス最適化部3より実行され、システム状態記録部24の内容を調べシステムの負荷状態を回答する。この例では、まず、システムの負荷状態として図8の最初のステップ401で計算されるアイドルCPU比率を求める。つぎに、システム状態記録部24内の各値を「0」にクリアする（ステップ402）。最後に上記で求めたアイドルCPU比率を問い合わせ元であるタイムスライス最適化部3に回答する（ステップ403）。

【0017】

【発明の効果】以上説明したように本発明は、一定期間毎にその間のシステム負荷状態により各プログラムのタイムスライス値を見直し、最適なシステム運用が行えるように自動的にタイムスライス値の調整を行うようにしたので、一定期間毎にシステムの負荷状況により自動的に各プログラムのタイムスライス値を調整することができ、容易に効率の良い最適なシステム運用ができるという効果がある。

# \*【図面の簡単な説明】

【図1】本発明の一実施例を示すブロック図である。

【図2】図1におけるタイムスライス情報格納部の構成例を示す説明図である。

【図3】図1におけるディスパッチ部のアルゴリズムの例を示すフローチャートである。

【図4】図1におけるタイムスライス最適化部のアルゴリズムの例を示すフローチャートである。

【図5】図1におけるタイムスライス最適化部で使用する調整値決定用の表の一例を示す説明図である。

【図6】図1におけるシステム負荷状態監視部の構成例を示すブロック図である。

【図7】図6におけるシステム状態調査部のアルゴリズムの例を示すフローチャートである。

【図8】図6におけるシステム状態回答部のアルゴリズムの例を示すフローチャートである。

【符号の説明】

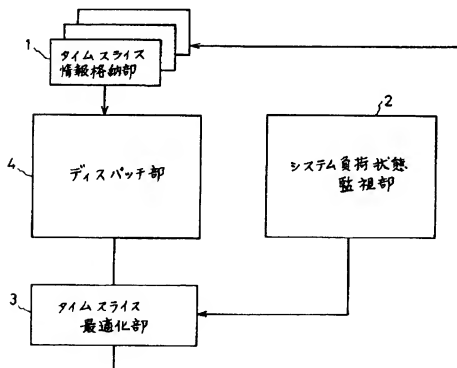
1 タイムスライス情報格納部

2 システム負荷状態監視部

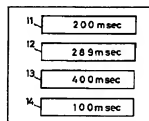
3 タイムスライス最適化部

4 ディスパッチ部

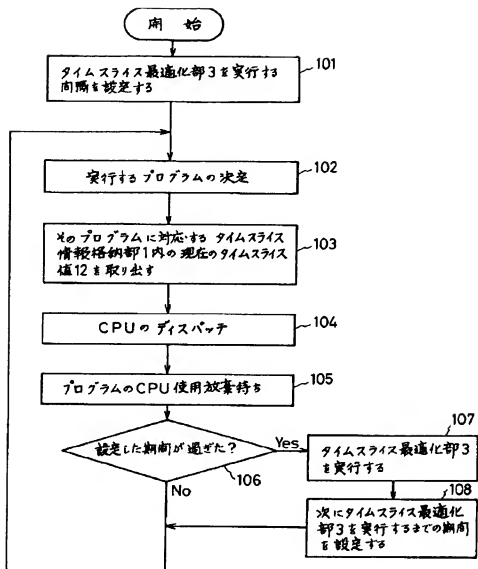
【図1】



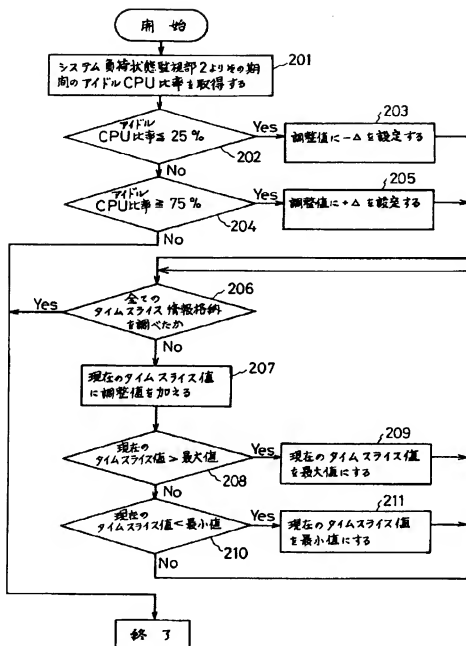
【図2】



【図3】



【図4】

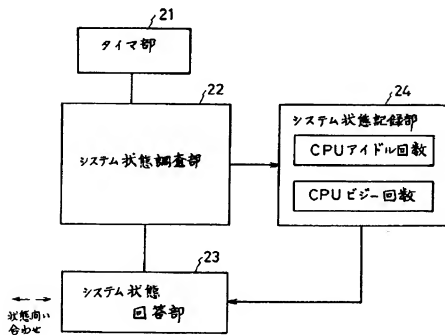




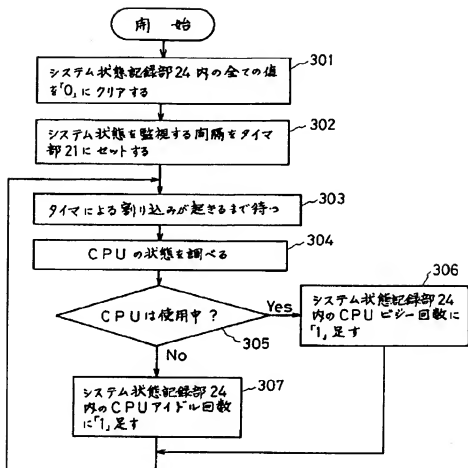
【図5】

条件部	調整値
0% ~ 10%	-10 msec
11% ~ 25%	-5 msec
26% ~ 75%	0 msec
76% ~ 90%	+5 msec
91% ~ 100%	+10 msec

【図6】



【図7】



【図8】

